嘟嘟独立博客

爱生活爱编码

search...

文章目录 隐藏目录

Spring Boot干货系列：（四）开发Web应用之Thymeleaf篇

关闭阅读模式

Spring Boot干货系列 Spring Boot 2017-03-13

前言

Web开发是我们平时开发中至关重要的，这里就来介绍一下Spring Boot对Web开发的支持。

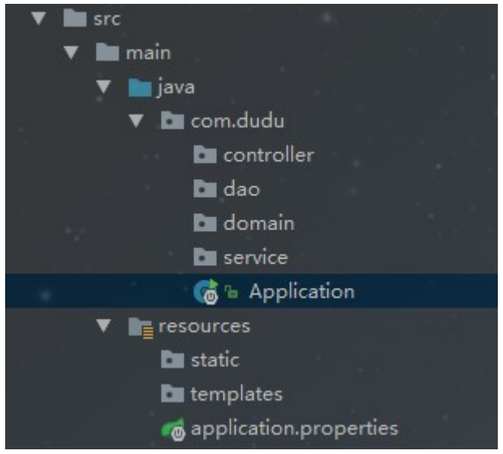
正文

Spring Boot提供了spring-boot-starter-web为Web开发予以支持，spring-boot-starter-web为我们提供了嵌入的Tomcat以及Spring MVC的依赖。

—— 项目结构推荐 ——

查看 3 条评论

一个好的项目结构会让你开发少一些问题，特别是Spring Boot中启动类要放在root package下面，我的web工程项目结构如下：



- root package结构：`com.dudu`
- 应用启动类`Application.java`置于root package下，这样使用@ComponentScan注解的时候默认就扫描当前所在类的package
- 实体（Entity）置于`com.dudu.domain`包下
- 逻辑层（Service）置于`com.dudu.service`包下
- controller层（web）置于`com.dudu.controller层`包下
- static可以用来存放静态资源
- templates用来存放默认的模板配置路径

—— Spring Web MVC框架介绍 ——

Spring Web MVC框架（通常简称为"Spring MVC"）是一个富"模型，视图，控制器"的web框架。

Spring MVC允许你创建特定的@Controller或@RestController beans来处理传入的HTTP请求。

示例：

```
1  @RestController
2  @RequestMapping(value="/users")
3  public class MyRestController {
4      @RequestMapping(value="/{user}", method=RequestMethod.GET)
5      public User getUser(@PathVariable Long user) {
6          // ...
7      }
8      @RequestMapping(value="/{user}/customers", method=RequestMethod.GET)
9      List<Customer> getUserCustomers(@PathVariable Long user) {
10         // ...
11     }
12     @RequestMapping(value="/{user}", method=RequestMethod.DELETE)
13     public User deleteUser(@PathVariable Long user) {
14         // ...
15     }
16 }
```

Spring MVC自动配置

Spring Boot为Spring MVC提供适用于多数应用的自动配置功能。在Spring默认基础上，自动配置添加了以下特性：

1. 引入ContentNegotiatingViewResolver和BeanNameViewResolver beans。
2. 对静态资源的支持，包括对WebJars的支持。
3. 自动注册Converter，GenericConverter，Formatter beans。
4. 对HttpMessageConverters的支持。
5. 自动注册MessageCodeResolver。
6. 对静态index.html的支持。
7. 对自定义Favicon的支持。

如果想全面控制Spring MVC，你可以添加自己的@Configuration，并使用@EnableWebMvc对其注解。如果想保留Spring Boot MVC的特性，并只是添加其他的MVC配置(拦截器，formatters，视图控制器等)，你可以添加自己的WebMvcConfigurerAdapter类型的@Bean（不使用@EnableWebMvc注解）,具体拦截器等配置后续文章会解析。

静态文件

默认情况下，Spring Boot从classpath下一个叫/static（/public，/resources或/META-INF/resources）的文件夹或从ServletContext根目录提供静态内容。这使用了Spring MVC的ResourceHttpRequestHandler，所以你可以通过添加自己的WebMvcConfigurerAdapter并覆写addResourceHandlers方法来改变这个行为（加载静态文件）。

在一个单独的web应用中，容器默认的servlet是开启的，如果Spring决定不处理某些请求，默认的servlet作为一个回退（降级）将从ServletContext根目录加载内容。大多数时候，这不会发生（除非你修改默认的MVC配置），因为Spring总能够通过DispatcherServlet处理请求。

此外，上述标准的静态资源位置有个例外情况是Webjars内容。任何在/webjars/**路径下的资源都将从jar文件中提供，只要它们以Webjars的格式打包。

注：如果你的应用将被打包成jar，那就不要使用src/main/webapp文件夹。尽管该文件夹是一个共同的标准，但它仅在打包成war的情况下起作用，并且如果产生一个jar，多数构建工具都会静悄悄的忽略它

模板引擎

Spring Boot支持多种模版引擎包括：

- FreeMarker
- Groovy
- Thymeleaf(官方推荐)
- Mustache

JSP技术Spring Boot官方是不推荐的，原因有三：

2. Jetty 嵌套的容器不支持jsp
3. Undertow
4. 创建自定义error.jsp页面不会覆盖错误处理的默认视图，而应该使用自定义错误页面

当你使用上述模板引擎中的任何一个，它们默认的模板配置路径为：`src/main/resources/templates`。当然也可以修改这个路径，具体如何修改，可在后续各模板引擎的配置属性中查询并修改。

Thymeleaf模板引擎

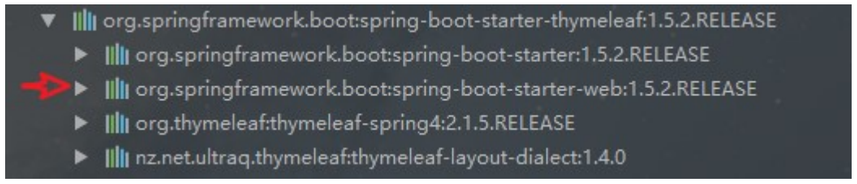
Thymeleaf是一款用于渲染XML/XHTML/HTML5内容的模板引擎。类似JSP，Velocity，FreeMaker等，它也可以轻易的与Spring MVC等Web框架进行集成作为Web应用的模板引擎。与其它模板引擎相比，Thymeleaf最大的特点是能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个Web应用。它的功能特性如下：

- Spring MVC中@Controller中的方法可以直接返回模板名称，接下来Thymeleaf模板引擎会自动进行渲染
- 模板中的表达式支持Spring表达式语言（Spring EL）
- 表单支持，并兼容Spring MVC的数据绑定与验证机制
- 国际化支持

Spring官方也推荐使用Thymeleaf,所以本篇代码整合就使用Thymeleaf来整合。

引入依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```



如图所示，spring-boot-starter-thymeleaf会自动包含spring-boot-starter-web，所以我们就不需要单独引入web依赖了。

编写controller

```
@Controller
@RequestMapping("/learn")
public class LearnResourceController {
    @RequestMapping("/")
    public ModelAndView index(){
        List<LearnResouce> learnList =new ArrayList<LearnResouce>();
        LearnResouce bean =new LearnResouce("官方参考文档","Spring Boot Reference Guide","http://docs
        learnList.add(bean);
```

```
        bean =new LearnResouce("龙国学院","Spring Boot 教程系列学习","http://www.roncoo.com/article/de
        learnList.add(bean);
        bean =new LearnResouce("嘟嘟MD独立博客","Spring Boot干货系列 ","http://tengj.top/");
        learnList.add(bean);
        bean =new LearnResouce("后端编程嘟","Spring Boot教程和视频 ","http://www.toutiao.com/m15590967
        learnList.add(bean);
        bean =new LearnResouce("程序猿DD","Spring Boot系列","http://www.roncoo.com/article/detail/12
        learnList.add(bean);
        bean =new LearnResouce("纯洁的微笑","Sping Boot系列文章","http://www.ityouknow.com/spring-boot
        learnList.add(bean);
        bean =new LearnResouce("CSDN——小当博客专栏","Sping Boot学习","http://blog.csdn.net/column/det
        learnList.add(bean);
        bean =new LearnResouce("梁桂钊的博客","Spring Boot 揭秘与实战","http://blog.csdn.net/column/de
        learnList.add(bean);
        bean =new LearnResouce("林祥纤博客系列","从零开始学Spring Boot ","http://412887952-qq-com.iteye
        learnList.add(bean);
        ModelAndView modelAndView = new ModelAndView("/index");
        modelAndView.addObject("learnList", learnList);
        return modelAndView;
    }
}
```

—— 编写html ——

引入依赖后就在默认的模板路径 `src/main/resources/templates` 下编写模板文件即可完成。这里我们新建一个index.html:

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>learn Resources</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<div style="text-align: center;margin:0 auto;width: 1000px; ">
<h1>学习资源大奉送，爱我就关注嘟嘟公众号：嘟爷java超神学堂（javaLearn）</h1>
<table width="100%" border="1" cellspacing="1" cellpadding="0">
    <tr>
        <td>作者</td>
        <td>教程名称</td>
        <td>地址</td>
    </tr>
    <!--/*@thymesVar id="learnList" type=""*/-->
    <tr th:each="learn : ${learnList}">
        <td th:text="${learn.author}">嘟嘟MD</td>
        <td th:text="${learn.title}">SPringBoot干货系列</td>
        <td><a th:href="${learn.url}" target="_blank">点我</a></td>
    </tr>
</table>
```

```
25 </body>
26 </html>
```

> 注：通过xmlns:th="http://www.thymeleaf.org" 命令空间，将静态页面转换为动态的视图，需要进行动态处理的元素将使用"th:"前缀。

ok,代码都写好了，让我们看对比下直接打开index.html和启动工程后访问http://localhost:8080/learn 看到的效果，Thymeleaf做到了不破坏HTML自身内容的数据逻辑分离。

学习资源大奉送，爱我就关注嘟嘟公众号：嘟爷java超神学堂
（javaLearn）

作者	教程名称	地址
嘟嘟MD	SPringBoot干货系列	点我

学习资源大奉送，爱我就关注嘟嘟公众号：嘟爷java超神学堂
（javaLearn）

作者	教程名称	地址
官方参考文档	Spring Boot Reference Guide	点我
官方SpriongBoot例子	官方SpriongBoot例子	点我
龙国学院	Spring Boot 教程系列学习	点我
嘟嘟MD独立博客	Spring Boot干货系列	点我
后端编程嘟	Spring Boot教程和视频	点我
程序猿DD	Spring Boot系列	点我
纯洁的微笑	Sping Boot系列文章	点我
CSDN——小当博客专栏	Sping Boot学习	点我
梁桂钊的博客	Spring Boot 揭秘与实战	点我
林祥纤博客系列	从零开始学Spring Boot	点我

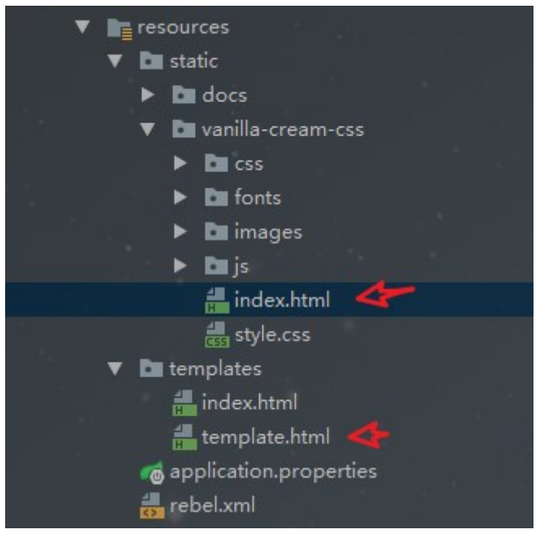
—— Thymeleaf的默认参数配置 ——

在application.properties中可以配置thymeleaf模板解析器属性

```
1  # THYMELEAF (ThymeleafAutoConfiguration)
2  #开启模板缓存（默认值: true)
3  spring.thymeleaf.cache=true
4  #Check that the template exists before rendering it.
5  spring.thymeleaf.check-template=true
6  #检查模板位置是否正确（默认值:true)
7  spring.thymeleaf.check-template-location=true
8  #Content-Type的值（默认值: text/html)
9  spring.thymeleaf.content-type=text/html
10 #开启MVC Thymeleaf视图解析（默认值: true)
11 spring.thymeleaf.enabled=true
12 #模板编码
```

```
#要被排除在解析之外的视图名称列表，用逗号分隔
spring.thymeleaf.excluded-view-names=
#要运用于模板之上的模板模式。另见StandardTemplate-ModeHandlers(默认值：HTML5)
spring.thymeleaf.mode=HTML5
#在构建URL时添加到视图名称前的前缀（默认值：classpath:/templates/）
spring.thymeleaf.prefix=classpath:/templates/
#在构建URL时添加到视图名称后的后缀（默认值：.html）
spring.thymeleaf.suffix=.html
#Thymeleaf模板解析器在解析器链中的顺序。默认情况下，它排第一位。顺序从1开始，只有在定义了额外的TemplateResolver B
spring.thymeleaf.template-resolver-order=
#可解析的视图名称列表，用逗号分隔
spring.thymeleaf.view-names=
```

—— 整合一个bootstrap框架给大家 ——



大家可以直接打开vanilla-cream-css下面的index.html来查看静态效果，如下：

动态效果的话可以查看template.html

这里把上面的资源例子重新用bootstrap写了下，效果不错哦，如下：

Sptring Boot学习资源大奉送，爱我就关注嘟嘟公众号：嘟爷java超神学堂

作者	教程名称	地址
官方参考文档	Spring Boot Reference Guide	点我
官方SpriongBoot例子	官方SpriongBoot例子	点我
龙国学院	Spring Boot 教程系列学习	点我
嘟嘟MD独立博客	Spring Boot干货系列	点我
后端编程嘟	Spring Boot教程和视频	点我
程序猿DD	Spring Boot系列	点我
纯洁的微笑	Sping Boot系列文章	点我
CSDN——小当博客专栏	Sping Boot学习	点我
梁桂钊的博客	Spring Boot 揭秘与实战	点我
林祥纤博客系列	从零开始学Spring Boot	点我

总结

本章到此就结束了，下一篇准备介绍下如何整合jsp,毕竟现在绝大多数的企业还是用jsp来作为模板引擎的。

想要查看更多Spring Boot干货教程,可前往：Spring Boot干货系列总纲

源码下载

(￣▽￣)↗[相关示例完整代码]

一直觉得自己写的不是技术，而是情怀，一篇篇文章是自己这一路走来的痕迹。靠专业技能的成功是最具可复制性的，希望我的这条路能让你少走